PGO: a Parallel Computing Platform for Global Optimization Based on Genetic Algorithm

Kejing He [a], Li Zheng [b,*], Shoubin Dong [a], Liqun Tang [c], Jianfeng Wu [d], Chunmiao Zheng [e]

[a] *Guangdong Key Laboratory of Computer Network, South China University of Technology, Guangzhou, China*

[b] *Center for Agricultural Resources Research, IGDB, Chinese Academy of Sciences, Shijiazhuang, China*

[c] *College of Traffic and Communications, South China University of Technology, Guangzhou, China*

[d] *Department of Earth Sciences, Nanjing University, Nanjing, China*

[e] *Department of Geological Sciences, University of Alabama, Tuscaloosa, AL, United States*

Abstract

This paper presents the design, architecture and implementation of a general parallel computing platform, termed PGO, based on the Genetic Algorithm for global optimization. PGO provides an efficient and easy-to-use framework for parallelizing the global optimization procedure for general scientific modeling and simulation processes. Along with a core optimization kernel built on a Genetic Algorithm, PGO also includes a general input generator and an output extractor that can facilitate its easy integration with various scientific computing tasks. In this paper, we demonstrate the efficiency and versatility of PGO with two different applications: (1) the parallelization of a large scale parameter estimation problem associated with modeling water flow in a heterogeneous deep vadose zone; (2) the parallelization of a complex simulation-optimization procedure for searching for an optimal groundwater remediation design. PGO is developed as an open source code, and is independent of the computer operating system. It has been tested in a heterogeneous computing environment consisting of Solaris 9, Fedora Core 2 Linux, and Microsoft Windows machines, and is freely available for download from http://grid.scut.edu.cn/PGO/.

Key words: parallel computing, high performance computation platform, global optimization, the Genetic Algorithm

1 Introduction

In scientific and engineering computing, optimization is one of the most frequently encountered computational problems. Optimization refers to the process of identifying values for unknown model parameters or control variables so that the given objective function can achieve its optimum. When the objective function is smooth and unimodal, local gradient-based optimization techniques (such as Newton and Quasi-Newton methods) are effective. Local gradient-based optimization methods have been widely used in many popular tools for the optimization process. For example, PEST (Doherty, 2004) uses a nonlinear optimization technique known as the Gauss-Marquardt-Levenberg method, and UCODE (Poeter and Hill, 1998) solves optimization problem by minimizing a weighted least squares objective function with respect to the unknown parameter values using a modified Gauss-Newton method. However, when objective function is discontinuous or characterized by many local optima, local search procedures will be less effective and optimization results will be very sensitive to the choices of initial values.

Many researchers have successfully applied global optimization methods, such as the Genetic Algorithm (GA), rather than local gradient-based optimization techniques to geoscientific computational problems. The Genetic Algorithm is an intelligent random search technique based on the theory of Darwinian evolution. It was first introduced by Holland (1975), and was later extended by De Jong (1985) and Goldberg (1989). In geoscientific computation, the GA has found applications in calibrating conceptual rainfall-runoff models (Wang, 1991), solving a multiple objective groundwater pollution containment problem (Ritzel et al., 1994), conducting the inverse modeling in hydrogeology (Karpouzos et al., 2001; Prasad and Rastogi, 2001), and optimizing the groundwater remediation and contamination control systems (Erickson et al., 2002; Bayer and Finkel, 2004).

However, GA tends to converge very slowly when being applied to a highly nonlinear external model with many parameters and subsequently an enormous search space. It may take weeks even months to accomplish an optimization task. Many have explored parallelization for speeding up GA computation. There are mainly three kinds of parallel Genetic Algorithms (PGAs): (1) the global single-population master-slave GAs; (2) the multiple-deme coarse-grained GAs; (3) the fine-grained (cellular) GAs. Detailed descriptions and analyses of various PGAs are given by Cantú-Paz (2000). Xu et al. (2002) used a fine-grained GA for the VLSI implementation of real-time GPS atti-

* Corresponding author. Tel.: +1-505-665-5745; fax: +1-505-665-5757.
Email address: lzheng@tmail.lanl.gov (Li Zheng, currently a visiting scientist at the Theoretical Division of the Los Alamos National Laboratory)

tude determination systems. Spataro et al. (2004) employed a master-slave GA to evolve a two dimensional cellular automata model for lava flow simulation. Cheng et al. (2005) used a hybrid method that combines coarse-grained GA (ring topology) with a fuzzy optimal model to calibrate rainfall-runoff models.

Despite many applications of GAs and PGAs in scientific research, their routine adoption for general modeling and simulations remains a non-trivial task. First, the modelers need interfaces for data exchange between the simulation model and the chosen GA tool. These include mapping model's unknown parameters/variables to GA's abstract chromosomes, generating model's input files, extracting results from model's output, and evaluating fitness. Second, while the design of GA is intrinsically suitable for the parallelization of its execution, the integration of a general model with a parallel computing environmental is however far from being straight forward. Modelers need to cope with the message transferring between different machines and the distributed data storage/access during optimization progress. When a complex application requires more than a single homogeneous cluster, the topologies and message transferring between heterogeneous clusters will be a complex job even for computer scientists.

This paper aims at presenting a general and easy-to-use parallel computing platform, termed PGO, based on the Genetic Algorithm for facilitating the global optimization in routine scientific computation. Along with a core optimization kernel built on GA and a parallel computing technique, PGO also includes a general input generator and an output extractor that can facilitate its easy integration with external models. The algorithm implemented in PGO belongs to the master-slave type PGA approach mentioned before. Using a loose-coupling approach to control and communicate with external models, PGO is highly flexible and scalable. PGO is programmed in Perl script language and developed as an open source code. It is independent of the computer operating system and has been tested in a heterogeneous computing environment consisting of Solaris 9, Fedora Core 2 Linux, and Microsoft Windows machines. PGO and its user manual are in public domain and available for download from http://grid.scut.edu.cn/PGO/.

This paper is organized as follows. Following the introduction in section 1, section 2 explains the theory of optimization, Genetic Algorithm, parallel technique, and their integration. Section 3 describes the architecture and configuration of PGO. In section 4, PGO is applied to (1) the parallelization of a large scale parameter estimation problem associated with modeling water flow in a heterogeneous deep vadose zone; and (2) the parallelization of a complex simulation-optimization procedure for searching for an optimal groundwater remediation design. Finally, some conclusions and discussions are made in section 5.

2 Optimization, Genetic Algorithm, and Parallel Technique

2.1 Problem Statement

Suppose there exists a general parametric mathematical model Ω with J inputs, K observable outputs and N variable parameters. The output $\vec{y}$ of Ω is:

$$\vec{y} = \Omega(\vec{x}, \vec{p}) \qquad \vec{x} \in \mathbb{R}^J, \vec{y} \in \mathbb{R}^K, \vec{p} \in \mathbb{R}^N \tag{1}$$

Then, the goal of optimization is to find suitable value $\vec{p}^*$ so that the objective function $\mathcal{F} : \mathbb{R}^K \to \mathbb{R}$ can achieve its optimum.

2.2 Nomenclature

N population size, number of chromosomes in each generation
P_e elite probability
P_c crossover probability
P_m mutation probability
L_t number of generations tolerated for no improvement on the objective before the GA is terminated.

2.3 The Genetic Algorithm used in PGO

When implementing the GA, each parameter p_i is represented by a real number called a gene. Genes are cascaded to form a longer string $\vec{p}$ named a chromosome. Chromosomes are used to represent the possible combinations of unknown parameter values. A collection of N chromosomes is called a population. In each generation, let $e_{i,j}$ be the fitness of the jth chromosome at the ith generation, the GA would search for the optimal fitness $e^i_{opt} = Optimal(e_{i,j})$ over the entire space of parameters and attempts to drive e^i_{opt} to optimum over the succeeding generations. Unlike the local gradient based methods, the Genetic Algorithm requires no calculation of the gradient and is not susceptible to the trapping of local minima.

Using the Genetic Algorithm for highly nonlinear optimization often requires considerable computational time, especially when a large number of unknown parameters are involved. The computational time required by the Genetic

Algorithm optimization can be roughly calculated by

$$Time = G \times N \times T \tag{2}$$

where G is the number of generations for achieving the convergence, N is the population size, and T is the time that a single run of the application model will take. For example, in the case of modeling water flow in deep vadose zone presented in section 4.1, the application of a standard GA will have a G of 600, N of 800 and T of 10 minutes, which amounts to a computational time of 9 years in total for achieving the optimal solutions. Such a computational cost is unacceptable in practice. Fortunately, the operations on individual chromosome are independent from each other within each generation, which makes the GA is inherently suitable for parallelization. In current study, we employ the parallel technique to distribute multiple chromosomes to parallel processors in order to speed up the computation. Meanwhile, we also adopt an elite GA which will allow the best chromosomes to survive into the next generation even though their probability being selected is small; The GA scheme used in PGO is illustrated in Fig. 1. Generally, elite selection reduces the number of generations and improves algorithm performance (De Jong, 1975; Goldberg, 1989). Further, we provide an interface in PGO for users to define their own

Step 1: Let G_t be the maximal number of generations allowed. Encode the parameters to be estimated and refer them as the chromosomes. Set $i = 0$, and $m = 0$.
Step 2: Initialize N chromosomes, let $i = 1$ and $m = 0$.
Step 3: Computational nodes decode the chromosomes and calculate the objective $e_{i,j}$ for every jth chromosome in the ith generation. Let $e^i_{opt} = Optimal_j(e_{i,j})$.
Step 4: If $e^i_{opt} = e^{i-1}_{opt}$, $m = m + 1$; otherwise, $m = 0$.
Step 5: If $m = L_t$ or $i = G_t$, terminate the algorithm; otherwise, pass $P_e \cdot N$ parents into next generation, mate N parents and generate $N - P_e \cdot N$ children, invoke mutation along with the crossover procedure.
Step 6: Set $i = i + 1$, go to step 3.

Fig. 1. The GA scheme used in PGO

decoding module based on existing knowledge.

Theoretically speaking, any real numbers in the allowed range could be eligible for being chosen as the parameter values. In a specific application, however, practitioners can specify a much narrower range for each parameter based on experiences or knowledge. PGO provides an interface to let users define their own decoding module (Fig. 2 and section 3.1). The decoding module is a function to map one space to another. For example, if the parameter values

are in $\mathbb{R}^N$ space, but the most reasonable combination of these parameter values can be classed into some categories, users can employ this interface to map category indicators to actual parameter values, and the search space will then decrease from $\mathbb{R}^N$ to $\mathbb{N}$. The decoding module can be configured in the configuration file (section 3.3). In the Genetic Algorithm search, a smaller search space will lead to a smaller population size and quicker convergence.

In the current version of PGO, we implement two kinds of selection algorithms. They are the roulette wheel and the binary tournament. Users can choose either one of them through the configuration file (section 3.3). We plan to include more selection algorithms (e.g. ranking) in the future version of PGO.

2.4 Coupling with Parallel Techniques

Since executing the decoding operation and evaluating the fitnesses of chromosomes in the same generation are independent of each other, we can improve the computational efficiency significantly by carrying out these operations (decoding and evaluation) on each chromosome in parallel rather than sequentially. In PGO, the parallel computing framework is organized as a global master-slave system (as categorized in PGA) and uses a central database management system (DBMS) for storing all the data during optimization progress. The server distributes chromosomes to computational nodes which are responsible for decoding the chromosome into real values, running the external application model, calculating fitness and returning fitness to the server.

As communication overhead can be neglected among independent model runs, a near-linear speedup with the numbers of processors used can be achieved. Although the MPI (Message Passing Interface) (Karonis et al., 2003) has been a popular choice in many parallel computing applications, we choose the central DBMS approach in PGO due to the following considerations. First, many modelers may not have access to Unix-like servers, and have to run their optimization jobs on windows-based PCs. Using MPI on windows may complicate the installation, configuration, and the use. Second, the central DBMS approach is capable of handling data exchange between master and slave machines. The well-organized data in DBMS are also valuable for further development. For example, the PGO with DBMS data structure is very suitable for integrating with a wider Grid environment to enable the service-oriented Grid computing (Foster and Kesselman, 1999). The Grid-enabled version of PGO is in fact currently under development in our laboratory. In the following sections, we will explain in details the architecture and configuration of this parallel computing platform.

3 The Software

3.1 Architecture

PGO consists of two parts: server module and computational module as illustrated in Fig. 2. The server module is mainly the optimization kernel which controls the evolution of whole optimization progress. The computational module runs on many computational nodes, each of which gets a chromosome from the server and decodes the abstract chromosome into real parameter values, and store them into template files so as to generate input files for external application model. When all the input files are ready, the computational module will run the external application model, extract the results from the output files produced by the model, evaluate the fitness and return the fitness to the server for further evolution.

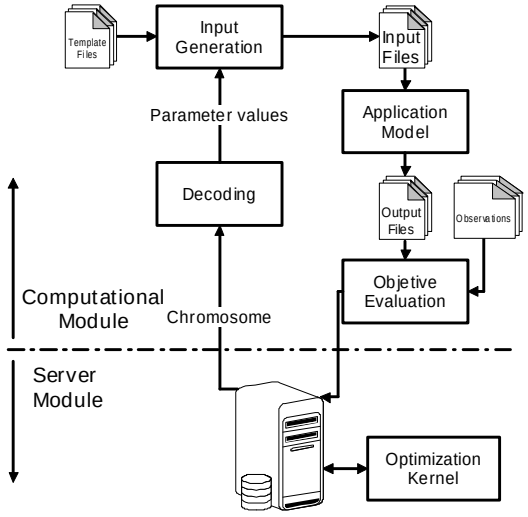

Fig. 2. Architecture of PGO. The server module controls the optimization progress; the computation module runs the external application model, returns the fitness to the server module for further evolution.

3.2 Integration

Since PGO has provided most components for a general optimization task, integrating PGO with a specific geoscientific modeling is as easy as installing and configurating a regular software. The steps are:

Step 1: Install prerequisite softwares, including Perl and the MySQL DBMS. PGO package also provides a script that make the installation of these softwares a very smooth process.

Step 2: Configurate PGO, edit the configuration file to suit your particular case. In section 3.3, we will explain the configuration process in detail.

Step 3: Copy the computational module to every computational nodes. Start the server module on the server, and start the computational module on each computational node.

3.3 Configuration

To facilitate the friendly-use of PGO, we provide a flexible XML file named config.xml to store all the configurations. Config.xml contains settings for the Genetic Algorithm employed (e.g. population size, crossover probability, and mutation probability etc.), the lower bound, upper bound and the precision of model parameters, rules for making model input files, extracting observations from model output files and evaluating the objective function. The graphical view of the XML schema used in PGO is illustrated in Fig. 3.

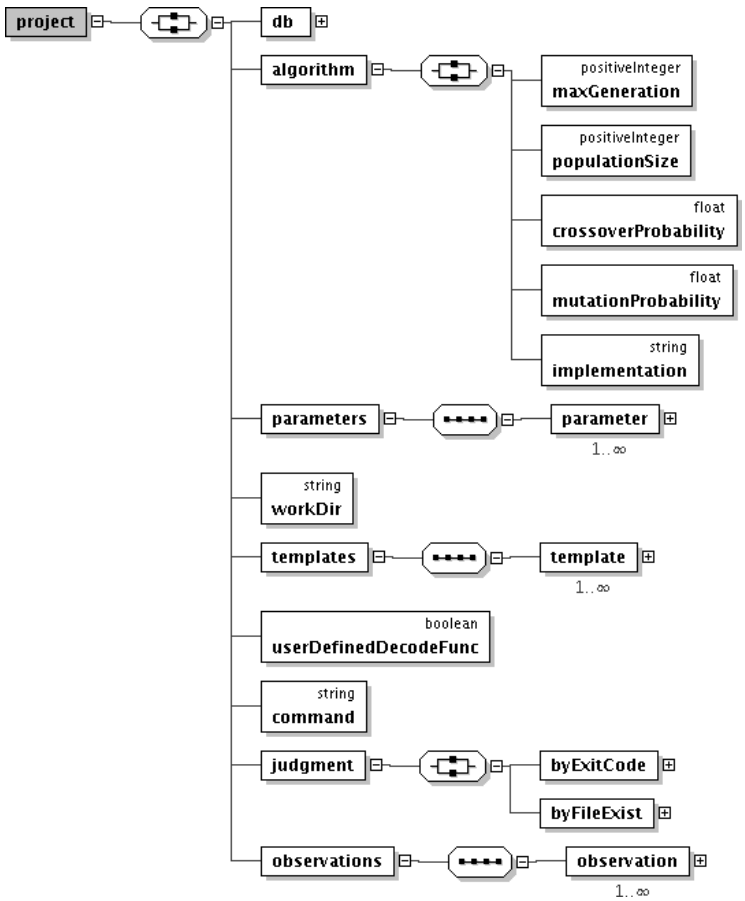

Fig. 3. The graphical view of the XML schema used for configuration

4 Applications

4.1 A large scale parameter estimation for modeling water flow in a heterogeneous deep vadose zone

The first application involves a simulation based on the SWAP model (Kroes and van Dam, 2003; Kendy et al., 2003) of the dynamic interactions between soil, water, atmosphere and plant in a deep vadose zone of a typical area in North China Plain, where years of intensive ground water pumping for irrigation have caused the rapid decline of ground water table. The one-dimensional SWAP model employs the Richard's equation, which combines Darcy's law and the continuity equation, to simulate the water flow process in the deep vadose zone. Based on the field observation, the soil column under study is divided into 25 layers with each layer having 7 unknown soil hydraulic property parameters as defined in the Van Genuchten relationship and Mualem's $K(\theta)$ function (Kroes and van Dam, 2003). We will then need to identify the 175 unknown parameters in order to perform the simulation. As we have explained earlier, the search for the best values for such a large number of parameters will be prohibitively expensive. Here we will apply PGO to this task to demonstrate the efficiency and utility of PGO.

Field data used for model inputs and calibration were collected at Luancheng Agricultural Eco-system Experimental Station of Chinese Academy of Sciences, Luancheng County, Hebei Province, from October 1998 to September 2001. The model input data include meteorological records, crop properties, and soil classifications. The data used for model calibration include the soil moisture contents as well as the ground water table elevations at the same sampling sites. The soil moisture contents were measured by neutron probes placed at nine depth intervals between 0 and 180 cm below ground surface every five days. Ground water table elevations were recorded by water level loggers at a same time interval. Detailed descriptions of experimental set up and procedures can be found in Liu et al. (2002).

The main input of SWAP is a *.swp file, which contains general section, crop section, soil water section etc. We assume that in this simulation all other parameters are fixed and known except the 175 soil hydraulic parameters contained in soil water section to be estimated. When conducting the parameter estimation by PGO, each computational node gets a chromosome from the server, decodes the chromosome into 175 soil hydraulic parameters and writes these values in a *.swp file. PGO has an interface available for users to specify the range of each parameter if desired. Then, it will run the SWAP model to produce a series of output files including a *.wba file for groundwater level data and a *.vap file for soil moisture content distribution. The computational

node will then extract values at the observation points from the *.wba file and the *.vap file, compare simulation results to corresponding measured values and return the fitness to the server. When the server has distributed all the chromosomes in current generation to all available computational nodes to accomplish all associated computations, the optimization kernel of PGO will breed next generation based on the scheme presented in Fig. 1. Iterating previous steps, PGO will drive fitness to maximum and obtain the optimal or near optimal parameter values. In current application, we set the maximal number of generations allowed (G_t) to 600. We also let L_t (section 2.2) equal to G_t so as to disable the termination of GA due to no improvement and allow the observation of entire optimization process. We carry out the parameter estimation task in 82 heterogeneous computational nodes with total 208 processors. With a population size (N) of 800 and the single run time of the application model (T) being about 10 minutes, the optimization process for 600 generations lasts half a month. Fig. 4 gives a sample comparison between the measured and model calculated soil moisture changes at the depth of 15cm below ground surface. Fig. 5 illustrates the convergence of RMSEs (Root Mean Square Errors) of soil moisture content and groundwater level during the parameter estimation process. As shown in Fig. 5, most of the convergence was achieved within 100 generations and there isn't any improvement in RMSE beyond 300 generations. So the algorithm in fact only takes about one week to converge. Of the two calibration criteria uses, the fitness between measured and model calculated soil moisture content dominated the optimization process. Considering the physics of the system under study, the ground water table fluctuation is much less sensitive than the soil moisture contents to the changing values of unknown soil hydraulic property parameters. Thus the water table fluctuation is much easier to match than soil water content. Over all, the RMSE between measured and model calculated water storage within top 180cm (the summation of water content over the 9 observation intervals) was 4.9cm (percentage error: 12.5%). The RMSE between measured and model calculated groundwater level was 1.44m (a pecentage error of 6.0%).

In the simulation of water flow in vadose zone, one common approach is to group the soil profile layers into a few soil types thus to dramatically reduce the number of soil hydraulic parameters needed to be estimated. In our application we preserved a large number of soil layers and assumed each layer is associated with a different set of unknown parameters. This approach maintains a higher degree of freedom and provides a more realistic representation for field sites with high heterogeneity. The use of PGO enables us to estimate exceedingly large number of unknown parameters in the search space. Without the adoption of distributed computing resources, this parameter estimation job as it is formulated may take the conventional GA up to 9 years to accomplish.

For comparison, we also applied a local gradient based search method, PEST

(Doherty, 2004), to this parameter estimation problem. The search process of PEST would stop quickly after it was started and couldn't move forward. Different stopping points were reached for different choices of initial values. Since there are 175 unknown parameters interacting with each other in a highly non-linear way, there exist a large number of local optima in the search space. It appeared that PEST was trapped in those local optima and failed to progress further.

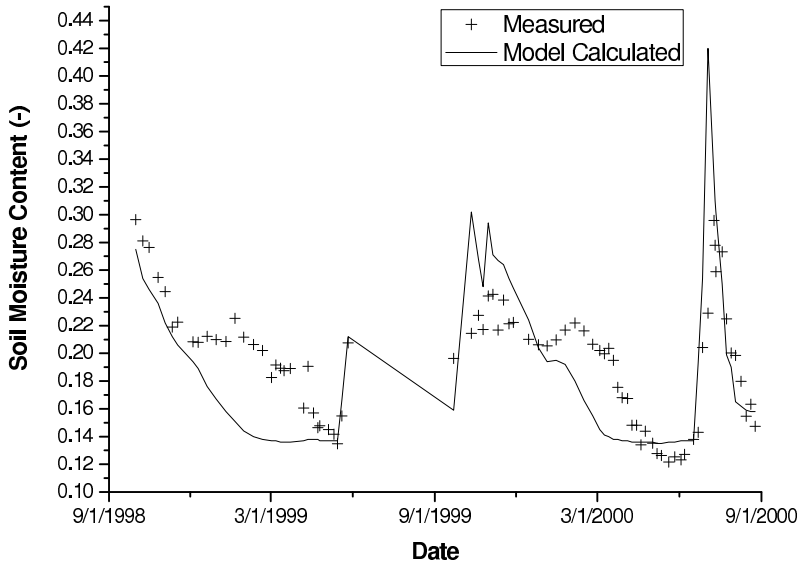

Fig. 4. Comparison between measured and model calculated soil moisture content at the depth of 15cm

4.2 Optimization of remediation system design

In our second application, we use PGO to implement a serial simulation-optimization code called 'Modular Groundwater Optimizer (MGO)' by Zheng and Wang (2003) to parallel environment. The MGO code couples the widely used groundwater flow simulator MODFLOW (McDonald and Harbaugh, 1988) and solute transport simulator MT3DMS (Zheng and Wang, 1999) with a general optimization package for formulating the most cost-effective groundwater remediation strategies under various physical, environmental and budgetary constraints. As PGO has been designed to be an easy-to-use general framework for parallel optimization, integrating PGO with MGO is simple and straightforward. Basically, the MGO code is modified to play the role of the computational module residing and running on individual nodes. Each run of MGO returns to the server a single objective function value associated with

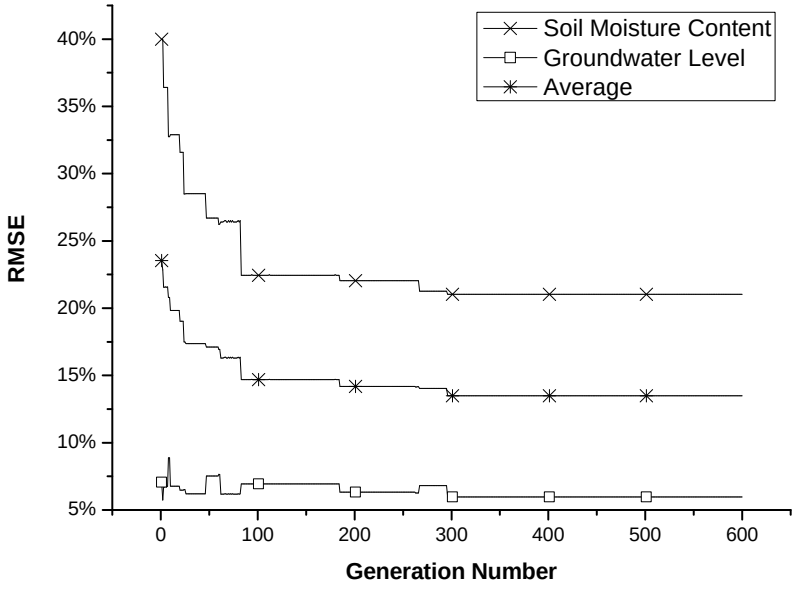

Fig. 5. Convergence of RMSEs in percentage during optimization process

each chromosome. The optimization algorithms in the original MGO code are no longer used; instead, the parallel GA kernel residing and running on the server carries out the optimization tasks.

We apply the parallelized MGO code to the optimization of a groundwater remediation system at the CS-10 site located within the Massachusetts Military Reservation (MMR) in Cape Cod, Massachusetts (Zheng and Wang, 2002). TCE is the primary contaminant at the CS-10 site. Extensive field sampling has led to the delineation of a TCE plume approximately 5 km long, 2 km wide, and up to approximately 43 m thick. The groundwater remediation system for the CS-10 site involved nine pumping wells intended to contain and eventually remove the TCE plume. The nine pumping wells must not exceed a total extraction rate approximately 10 m^3/min, the design capacity for the on-site treatment plant. After the extracted water was treated at the on-site treatment plant, it would be re-injected into the infiltration trenches on the downstream edges of the plume. More detailed information on the case study can be found in Zheng and Wang (2002).

Using the MMR example as a demonstration for the new parallel MGO for comparison with the original serial MGO, we optimize the pumping rates of the nine wells to achieve the maximum amount of contaminant mass removal by the nine managed wells and the well fence along Sandwich Road by the end of the project horizon while satisfying all the constraints discussed in Zheng and Wang (2002). In this application, the maximal number of generations allowed (G_t) is 50, L_t is 20 (defined in section 2.2), the population size (N)

is 200 and the run time (T) is about 25 minutes for running both flow and transport simulations on a single processor. We carry out the optimization in 36 heterogeneous computational nodes, with 72 processors in total. The optimization takes 52 hours to converge. The maximum amount of contaminant mass removal by the nine managed wells and the well fence along Sandwich Road is 3864 kg. In the original case, the formulation used is to maximize the mass removal by the managed wells only, and the result is 2864kg (Zheng and Wang, 2002). Adopting PGO, the reduction in computing time is proportional to the number of processors employed. Fig. 6 illustrates the evolution of objective function (contaminant mass removed) with generations.

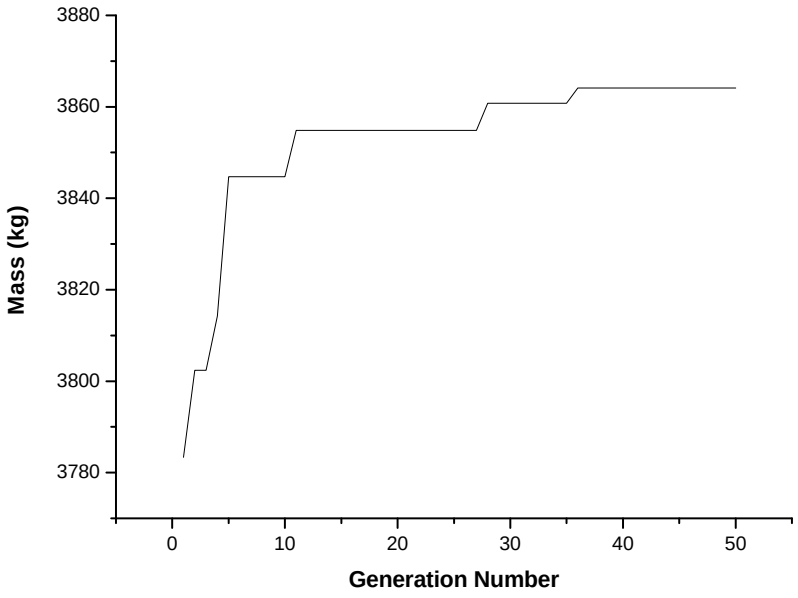

Fig. 6. Relationship between contaminant mass removal and generation number

5 Conclusions

This paper has presented the design, architecture and implementation of a general parallel framework (PGO) for GA-based global optimization. The design and architecture of PGO are highly flexible and scalable. The parallel optimization kernel is built on a master-slave type GA approach. Along with this kernel, we have constructed multiple interfaces to facilitate the easy integration of PGO with existing simulation models. These interfaces include input file generator, output extractor, fitness evaluator, and parameter bound settings etc. To simplify the use of PGO, we have employed a flexible XML file to store all the project settings and configurations. Furthermore, we adopted

the DBMS data structure, which facilitates the use of PGO in heterogeneous computing environment, including Unix, Linux and Window based computers. The use of DBMS data structure is also very suitable for integrating with a wider Grid environment to enable the Service-Oriented Grid computing and gain more powerful computer resources. PGO is programmed in Perl script language and developed as an open source code. Users may choose to expand and further develop this platform to meet their own requirements. The executable files, source codes, and the user manual are freely available for download from http://grid.scut.edu.cn/PGO/.

In this paper, we applied PGO to two case studies to illustrate the effectiveness and efficiency of PGO. In the first application case, we demonstrated that PGO could be easily used to control and run the external simulation model to obtain optimal parameter estimations without performing any recoding on the existing water flow model. The loose-coupling between PGO and external models can be easily achieved via the input generation interface and output extraction interface of PGO. In the second application, we showed that PGO can be used to easily parallelize an existing optimization code (MGO). With a few small modifications, PGO would take over the place of the GA optimization module within MGO, and be able to run the unchanged external ground water flow and transport model in parallel to optimize the remediation design. With the loose-coupling approach, the communication time between PGO and external models is often negligible as compared to the time required to run external models. The progress of entire optimization process after employing PGO would achieve almost linear speedup.

In future, we are planning to implement more selection methods (e.g. ranking) to give users more choices. The Grid-enabled version of PGO is also a focus of our further development. A paper on the preliminary results of Grid-enabled PGO is available in our download site. In addition to the global master-slave PGA adopted here, we will also explore the possibilities of including other kinds of PGAs, such as coarse-grained GA or fine-grained GA into the PGO platform.

Acknowledgements

This research was jointly funded by the Innovation Knowledge Project of Chinese Academy of Sciences (project No. KZCX3-SW-428), the ChinaGrid Project (project No. CG2003-GA002 and CG2003-GA005), and Natural Science Foundation of China (NSFC) (grant No. 40472130). Li Zheng's research was partially supported by the Los Alamos National Laboratory LDRD Project "High-Resolution Physically-Based Model of Semi-Arid River Basin Hydrology" and in collaboration with SAHRA Program of the National Science Foun-

dation under Agreement No. EAR-9876800, and by the US Department of Energy Office of Science's Advanced Scientific Computing Research (ASCR) Applied Mathematical Research program. We greatly appreciate the Luancheng Agricultural Eco-system Experimental Station of Chinese Academy of Sciences for making available the field data used in this study. Also many thanks to Joop Kroes (Alterra, Wageningen University and Research Centre) for supporting the source code of TTUTIL library (van Kraalingen and Rappoldt, 2000).

References

Bayer, P., Finkel, M., 2004. Evolutionary algorithms for the optimization of advective control of contaminated aquifer zones. Water Resources Research 40 (6), w06506, doi:10.1029/2003WR002675.

Cantú-Paz, E., 2000. Efficient and Accurate Parallel Genetic Algorithms. Kluwer Academic Publishers, Norwell, MA, USA.

Cheng, C., Wu, X., Chau, K. W., 2005. Multiple criteria rainfall-runoff model calibration using a parallel genetic algorithm in a cluster of computers. Hydrological sciences journal 50 (6), 1069 – 1087.

De Jong, K. A., 1975. An analysis of the behavior of a class of genetic adaptive systems. Ph.D. thesis, University of Michigan.

De Jong, K. A., 1985. Genetic algorithms: A 10 year perspective. In: Grefenstette, J. J. (Ed.), ICGA. Lawrence Erlbaum Associates, pp. 169–177.

Doherty, J., 2004. PEST: Model-Independent Parameter Estimation. Watermark Numerical Computing, 5th Edition.

Erickson, M., Mayer, A., Horn, J., 2002. Multi-objective optimal design of groundwater remediation systems: application of the niched Pareto genetic algorithm (NPGA). Advances in Water Resources 25 (1), 51–65.

Foster, I., Kesselman, C. (Eds.), 1999. The grid: blueprint for a new computing infrastructure. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Goldberg, D. E., 1989. Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, Reading, Mass.

Holland, J., 1975. Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor, MI.

Karonis, N. T., Toonen, B., Foster, I., 2003. MPICH-G2: A grid-enabled implementation of the message passing interface. Journal of Parallel and Distributed Computing 63 (5), 551–563.

Karpouzos, D. K., Delay, F., Katsifarakis, K. L., de Marsily, G., 2001. A multipopulation genetic algorithm to solve the inverse problem in hydrogeology. Water Resources Research 37 (9), 2291–2302.

Kendy, E., Gérard-Marchant, P., Walter, M. T., Zhang, Y., Liu, C., Steenhuis, T. S., 2003. A soil-water-balance approach to quantifying groundwater recharge from irrigated cropland in the north china plain. Hydrol. Process.

17, 2011–2031.

Kroes, J., van Dam, J., 2003. Reference Manual SWAP version 3.0.3. Alterra, Green World Research, Wageningen, The Netherlands.

Liu, C., Zhang, X., Zhang, Y., 2002. Determination of daily evaporation and evapotranspiration of winter wheat and maize by large-scale weighing lysimeter and micro-lysimeter. Agricultural and Forest Meteorology 111 (2), 109–120.

McDonald, M. G., Harbaugh, W. W., 1988. A modular three-dimensional finite-difference ground water flow model. Techniques of Water-Resources Investigations, Book 6. US Geological Survey, Reston, VA, Ch. A1.

Michalewicz, Z., 1996. Genetic Algorithms + Data Structures = Evolution Programs, 3rd Edition. Springer Verlag.

Poeter, E. P., Hill, M. C., 1998. Documentation of UCODE: A computer code for universal inverse modeling. Water-Resources Investigations Reports 98-4080, U.S. Geological Survey.

Prasad, K. L., Rastogi, A. K., 2001. Estimating net aquifer recharge and zonal hydraulic conductivity values for Mahi Right Bank Canal project area, India by genetic algorithm. Journal of Hydrology 243, 149–161.

Ritzel, B. J., Eheart, J. W., Ranjithan, S., 1994. Using genetic algorithms to solve a multiple objective groundwater pollution containment problem. Water Resources Research 30 (5), 1589–1603.

Spataro, W., D'Ambrosio, D., Rongo, R., Trunfio, G. A., 2004. An evolutionary approach for modelling lava flows through cellular automata. In: Sloot, P. M. A., Chopard, B., Hoekstra, A. G. (Eds.), ACRI. Vol. 3305 of Lecture Notes in Computer Science. Springer, pp. 725–734.

van Kraalingen, D., Rappoldt, C., 2000. Reference manual of the FORTRAN utility library TTUTIL v. 4. Plant Research International, Wageningen, The Netherlands.

Wang, Q. J., 1991. The genetic algorithm and its application to calibrating conceptual rainfall-runoff models. Water Resources Research 27 (9), 2467–2471.

Xu, J., Arslan, T., Wang, Q., Wan, D., 2002. An ehw architecture for real-time gps attitude determination based on parallel genetic algorithm. In: 2002 NASA/DoD Conference on Evolvable Hardware, 15-18 July 2002, Alexandria, VA, USA. Vol. 2002. Los Alamitos, CA, USA : IEEE Comput. Soc, 2002, pp. 133 – 141.

Zheng, C., Wang, P. P., 1999. MT3DMS: Documentation and user's guide. Contract report SERDP-99-1, U.S. Army Eng, R&D Center, Vicksburg, MS.

Zheng, C., Wang, P. P., 2002. A field demonstration of the simulation-optimization approach for remediation system design. Ground Water 40 (3), 258–265.

Zheng, C., Wang, P. P., May 2003. MGO-A Modular Groundwater Optimizer Incorporating MODFLOW/MT3DMS. University of Alabama and Groundwater Systems Research Ltd.